پروژه ی پایانی درس گفتار پردازی رقمی  
نام : حسین سیم چی، علی یزدانی  
98443119 (سیم چی)، 98443257 (یزدانی)  
استاد : آقای دکتر یاسر شکفته

14/11/1399

**توضحات مربوط به فایل گزارش و پروژه :**

پروژه ی نوشته شده شامل 3 قسمت است

1. قسمت اول شامل مقدمه، بررسی توابع استخراج ویژگی و کاربرد هریک از آن ها جهت تحلیل احساسات و همچنین بررسی خروجی **اولین کد نوشته شده** جهت دسته بندی احساسات موجود در یک جمله می باشد ( در این کد قسمتی نیز به عنوان Prediction درنظر گرفته شده که قادر خواهد بود برای یک فایل صوتی دلخواه، احساس مربوط در آن را تشخیص دهد )
2. قسمت دوم شامل **دومین کد نوشته شده**، توضیح کامل آن و آموزش کامل بر روی دیتاست “ShEMO” . ( دارای تکنیک های مختلف یادگیری ماشین و عمیق و همچنین استفاده ی بیشتری ویژگی نسبت به کد اولی است )
3. قسمت سوم، شامل توضیح نحوه ی اجرا شدن اولین کد بر روی دیتاست های “Berlin” و “ShEMO” به فرمت **مقاله**

کد اول و دوم به واسطه ی تعدد ویژگی ها و همچنین به لحاظ سادگی و دقت تفاوت هایی باهم دارند. کد دوم درنظر گرفته شده 46 ویژگی را استخراج می کند درحالی که این عدد در کد اول کمتر بوده و 36 ویژگی خواهد بود.

**قسمت اول**

**کد اول**

در این قسمت به بررسی توابع نوشته شده یا کلاس های مورد نیاز و همچنین مدل نوشته شده جهت دسته بندی احساسات می پردازیم. همچنین در قسمت دوم، سعی شده است مراحل انجام کد و نحوه ی اجرای آن را بررسی کنیم.

در این پروژه هدف پیاده سازی مقاله ی تشخیص احساسات ارائه شده می باشد به طوری که جهت بررسی بیشتر سعی شده است بر روی دو دیتاست دیگر از قبیل “ Berlin “ و “ShEMO” خروجی گرفته شود.

**\*\*\* تصاویر بدست آمده از خروجی پروژه و مدل نوشته در طول توضیح گزارش قرار داده شده است.**

**مقدمه**

تشخیص احساسات یکی از چالشی ترین موضوعات مورد بررسی متخصصان هوش مصنوعی بوده است به طوری که در بسیاری از کاربردها مانند بازی های کامپیوتری، رباتها و ارتباط انسان ها، احساسات نقش بسیار مهمی دارند. یکی از راه های تشخیص احساس، استفاده از سیگنال گفتار و ارزیابی ویژگی هایی است که به بهتر شناسایی شدن احساس موجود در گفتار کمک می کند. با گسترش استفاده از روش های یادگیری عمیق در سال های گذشته، در این پروژه سعی شده است یکی از پراستنادترین روش های چندسال گذشته جهت تشخیص احساسات را استفاده کنیم. روش مورد نظر که با نام Bidirectional LSTM with Local Attention شناخته شده است در ابتدا هر جمله را به فریم هایی مجزا تقسیم میکند و سپس سعی میکند با توجه به واکدار بودن، بیواک بودن و یا ساکت بودن هر فریم وزنی را به آن ها احتصاص داده تا فریمی که دارای کلمات با احساس بیشتری است وزن بیشتری را داشته باشد *و لذا در نهایت تاثیر بیشتری نیز بر روی به روزرسانی وزن ها داشته باشد*. در این پروژه هدف پیاده سازی مقاله گفته شده و سپس ارزیابی بر روی چند دیتاست معروف از قبیل “Berlin” و “ShEMO” است. تمامی نتایج و قسمت های مختلف انجام پروژه به صورت گزارش در ادامه آمده شده است. در این قسمت سعی می کنیم توابع و مدل نوشته شده را با دقت بررسی و تحلیل کنیم و در قسمت دوم نیز مراحلی که کد طی می کند تا خروجی را نشان دهد، بررسی می کنیم.

**\*\*\* تمامی فایل ها، تصاویر و مدل های نوشته شده در فایل گزارش ارسال و تقدیم می گردد.**

**توابع نوشته شده جهت استخراج ویژگی ( گام اول )**

**کلاس دیتاست**

جهت برچسب گذاری داده های درون دیتاست از آن استفاده شده است.

دو پارامتر مهم که در زمان ساخت شی از کلاس باید به آن داده شوند، پارامترهای اسم دیتاست و مسیری است که دیتاست در آن قرار داده شده است.

**\*\*\* از کتابخانه ی Librosa جهت خواندن فایل های صوتی و بدست آوردن فرکانس نمونه برداری و سیگنال در طول پروژه استفاده شده است.**

هر فایل صوتی که در دیتاست قرار داده شده است دارای یک نام است که خود دارای برچسب موردنظر برای داده است. به عنوان مثال برای دیتاست “Berlin” حرف پنجم و در دیتاست “ShEMO” حرف چهارم دارای برچسب مخصوص به داده است که با برخی از واژگان انگلیسی این کلاس ها مشخص شده اند.

درنهایت خروجی این کلاس شامل 3 جز است

1. کلاس استخراج شده برای هر داده
2. سیگنال مربوط به فایل صوتی مورد نظر
3. فرکانس نمونه برداری شده ی سیگنال

**تابع استخراج ویژگی Chroma ( stChromaFeaturesInit )**

از این ویژگی برای بررسی فایل های صوتی مربوط به موسیقی استفاده می شود و به طور کلی برای کلاس بندی Pitch ها مورد استفاده قرار می گیرد، به این صورت که فرض می کنیم Pitch ها می توانند دارای 12 نوع کلاس باشند. از این ویژگی ها بر اساس تحقیقات انجام شده **برای شناسایی بلندی یا آرامی صدا استفاده می شود** که در برخی از کابردهای شناسایی احساسات مثل شناسایی عصبانی بودن مخاطب مورد استفاده قرار می گیرد.

**تابع محاسبه ی فیلتربانک برای بدست آوردن ویژگی MFCC ( mfccInitFilterBanks )**

برای محاسبه ی ویژگی MFCC در ابتدا نیاز داریم تا سیگنال خود را به فضای فرکانس برده و سپس با اعمال تعداد فیلتربانک های مشخص بر روی آن و بقیه ی مراحل، این مقدار را بدست آوریم. ( در این پروژه تعداد فیلتر ها برابر 40 درنظر گرفته شده است )

مراحل استفاده شده جهت اعمال فیلتربانک

1. مقدار دهی های لازم ( تعداد فیلتر های برابر 40 )، فیلترهای اعمال شده مثلثی هستند.
2. بدست آوردن فرکانس نقاطی که در داخل یک فیلتر قرار می گیرند
3. محاسبه ی ضرایب فیلتربانک

درنهایت سیگنال فیلتر شده و ضرایب را برمی گردانیم.

\*\*\* از توابع و فرمول های موجود در اینترنت جهت استخراج ویژگی هایی که نام می بریم استفاده شده است.

**مراحل بدست آوردن ویژگی MFCC**

همانطور که فرآیند فوق مشخص است، برای بدست آوردن ویژگی MFCC باید 4 مرحله ی فوق را طی کنیم که در این پروژه هر 4 مرحله به صورت توابع نوشته شده است

**تابع استخراج ویژگی Zero Crossing Rate ( stZCR )**

از ویژگی ZCR برای **Voice activity detection** استفاده می شود که منجر می شود تشخیص دهیم که آیا در فریم موردنظر گوینده حضور دارد یا خیر. با استفاده از این ویژگی می توانیم فریم های ساکت را از بقیه ی فریم ها جدا کنیم

**تابع استخراج ویژگی انرژی ( stEnergy )**

از ویژگی انرژی سیگنال هم می توان بر دسته بندی فریم های ساکت، بی واک و واکدار استفاده نمود. چراکه می دانیم فریم هایی که دارای انرژی زیادی هستند، کلمات و جملات بیشتری را شامل شده اند.

**تابع استخراج ویژگی آنتروپی انرژی ( stEnergyEntropy )**

از ویژگی آنتروپی انرژی نیز برای **Voice activity detection** استفاده می شود. درحالت کلی از ویژگی آنتروپی (stSpectralEntropy ) جهت تشخیص تعداد پیک های یک سیگنال در یک فریم استفاده می شود.

**تابع استخراج ویژگی مرکز طیف سیگنال ( stSpectralCentroidAndSpread )**

مرکز طیف جایی است که در فضای Spectral دارای شدت روشنایی زیادی است و از این ویژگی می توانیم استفاده کنیم تا مقدار Spread را بدست آوریم. **از مقدار Spread برای مشخص کردن نویزی بودن یا نبودن سیگنال می توان استفاده کرد. یعنی اگر سیگنالی دارای Spread زیادی باشد بدین معنا است که نویزی است.**

**تابع استخراج ویژگی Flux ( stSpectralFlux )**

از ویژگی Flux می توان برای محاسبه ی میزان سرعت تغییر سیگنال استفاده نمود. برای این منظور باید فریم فعلی را با قبلی مقایسه کنیم. از کاربردهای این ویژگی می توان به تشخیص شروع یک جمله یا کلمه و میزان صدای سیگنال اشاره کرد.

**تابع استخراج ویژگی Rolloff ( stSpectralRollOff )**

از این ویژگی نیز در برخی موارد برای تشخیص صداهای نویزی استفاده می شود. به این صورت سیگنالی که از مقدار Rolloff پایین تر باشد به عنوان صدای نویزی تشخیص داده می شود.

**تابع محاسبه ی ماتریس DCT برای بدست آوردن MFCC**

برای بدست آوردن ویژگی MFCC مطابق آنچه در کلاس توضیح داده شد نیازی داریم تا ماتریس DCT را بدست آوریم که توابع آن ها بدست آورده شده است.

**تابع استخراج ویژگی ( stFeatureExtraction )**

کار این تابع استخراج تمامی ویژگی های نام برده شده و ذخیره در فایلی جداگانه است تا در زمان آموزش از آن استفاده کنیم.

**مدل نوشته شده جهت آموزش و دسته بندی احساسات موجود در جملات ( گام دوم )**

1. تابع نوشته شده جهت تقسیم دیتاست به دو قسمت آموزش و تست و همچنین مشخص کردن پارامترهایی جهت بدست آوردن مکانیزم Attention ( get\_data Function )
2. تابع Create\_model که دربردارنده ی هسته ی اصلی مدل، یعنی جایی که مدل را تعریف می کنیم تا در ادامه با استفاده از داده های آموزش و تست بر روی آن خروجی بگیریم.

**تابع get\_data**

در مراحل قبل با استفاده از کلاس دیتاست و توابع استخراج ویژگی که در مورد آن ها صحبت کردیم و همچنین طبق روالی که در قسمت دوم گزارش به آن خواهیم پرداخت، داده های استخراج شده و برگه ای که مربوط به استخراج ویژگی های متناظر می باشد و اطلاعات هر داده که نشان دهنده ی تعداد ویژگی های استخراج شده از آن داده میباشد را باید Load کنیم. سپس باید با استفاده از تابع train\_test\_split داده های موجود را به دو بخش آموزش و تست تقسیم کنیم.

برای داده های آموزش و تست، باید مقداری را به بردارهای آن ها الحاق کنیم. و در نهایت با ضرب نظیر به نظیر ورودی در خروجی و عبور از تابع فعال ساز Softmax، مکانیزم Attention را لحاظ می کنیم. بردارهایی که باید به داده اضافه کنیم را در پروژه u\_train و u\_test نامیدیم که بردارهایی به اندازه ی بردارهای train و test هستند و مقادیر آن ها یک تقسیم بر 256 درنظر گرفته شده است درنتیجه بردار u مانند زیر خواهد بود:

ساختار شبکه در زیر آمده است که توضیحات آن نوشته شده است

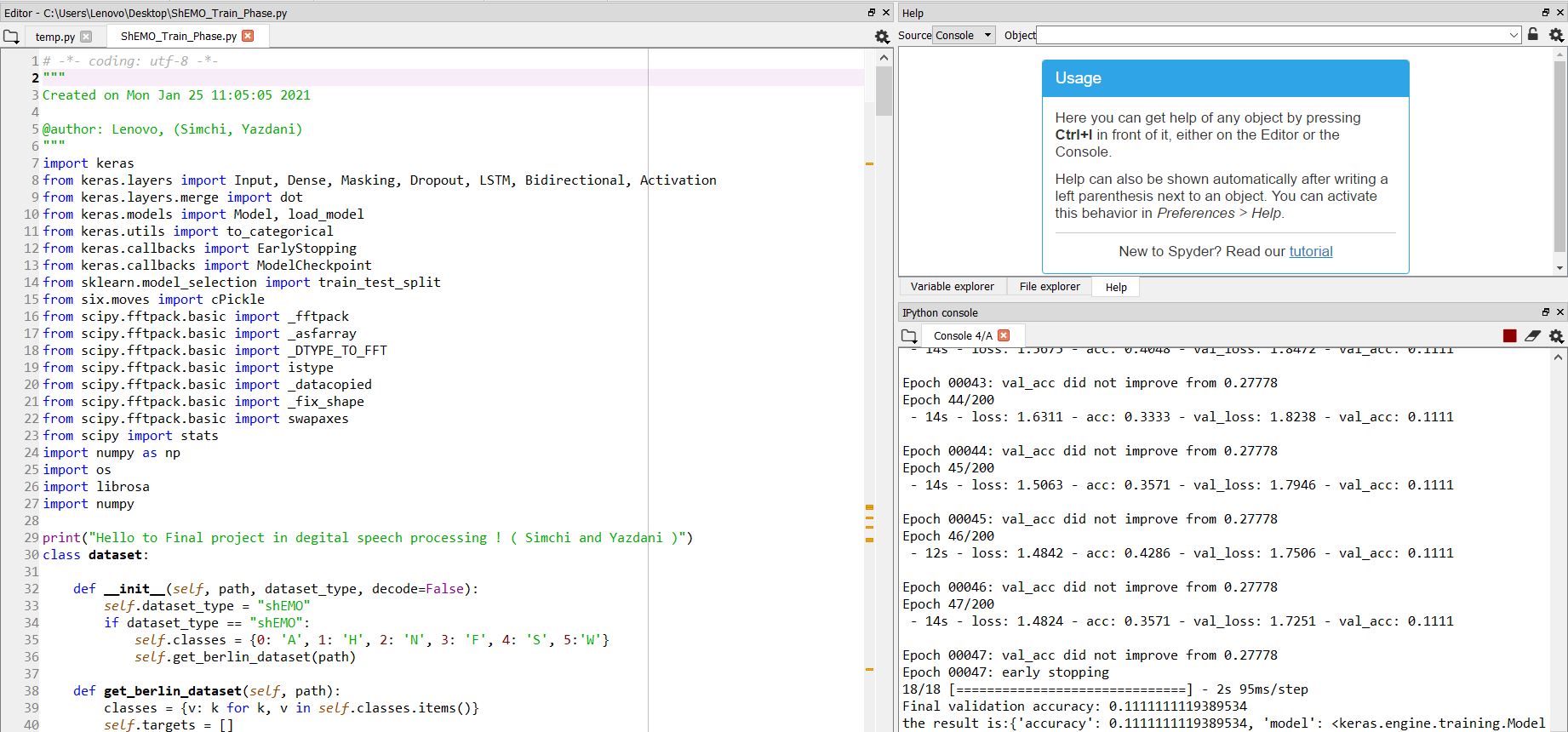
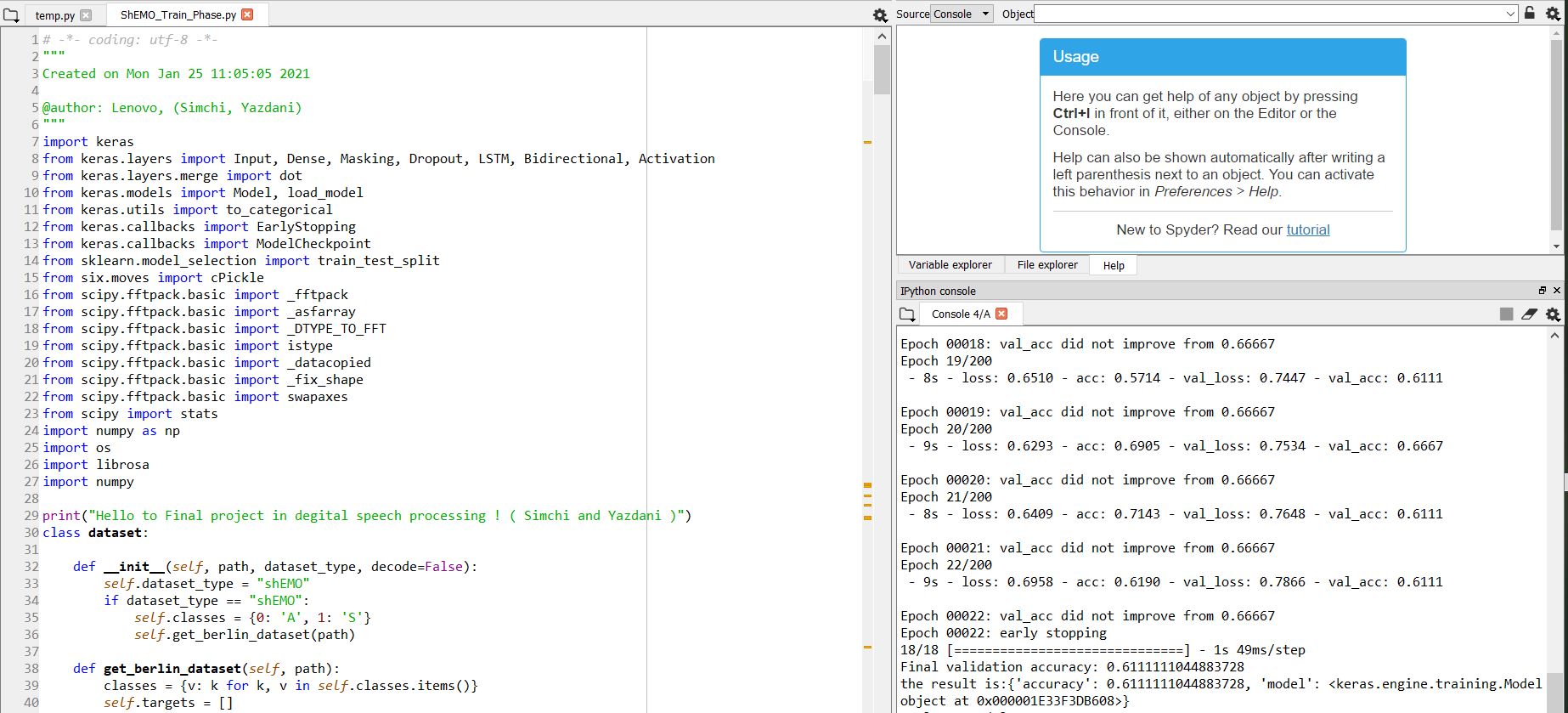
1. Input attention >>> دو مرحله ی اول و دوم صرفا برای مقدار دهی وزن ها است و تاثیری بر لایه های بعدی ندارد
2. Dense layer 1
3. Input features
4. Masking layer >>> از این لایه استفاده می شود تا بگوییم که توالی زمانی بین داده های ورودی نداریم
5. Dense layer 2
6. Dropout layer 1
7. Dense layer 3
8. Dropout layer 2
9. Bidirectional LSTM layer
10. Dot ( num2 and num9 ) >>> خروجی لایه ی 9 را در بردار وزن های اولیه ضرب می کنیم
11. Activation Softmax >>> مقادیر به صورت احتمالاتی بین صفر و یک قرار میگیرند
12. Dot ( num11 and num9 ) >>> با ضرب لایه ی 9 در 11، خروجی به صورت وزن دارد خواهد بود
13. Output layer ( Dense layer 4 )

مدل نهایی به صورت زیر خواهد بود

model = Model(inputs=[input\_attention, input\_feature], outputs=output)

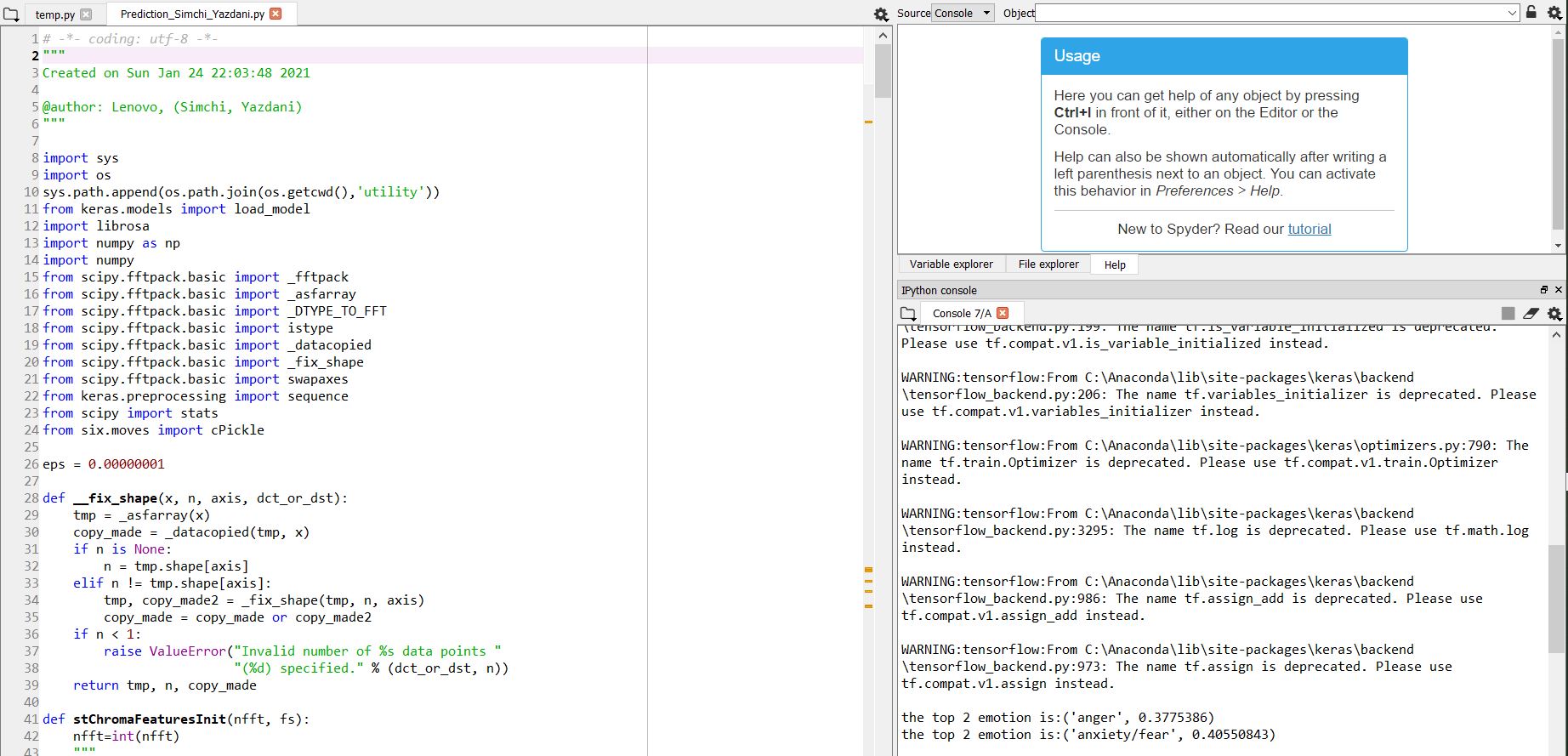
**نتایج بدست آمده ( اولین کد ) بر روی دیتاست “ShEMO” و “Berlin”**

یکی از مشکلات موجود برای آموزش و سپس تست در شبکه های عمیق برای اهداف پردازش سیگنال نیاز مبرم به حافظه ی زیاد جهت پردازش می باشد. درنتیجه به دلیل مشکلات سخت افزاری قادر به گرفتن خروجی بر روی کل دیتاست نبوده ایم و خروجی را بر روی تنها 60 داده ی صوتی گرفته ایم. فایل دیتاست استفاده شده دارای صدای هر دو جنس مرد و زن میباشد که جهت افزایش دقت و پیش بینی، از هر دو صدا استفاده شده است.

1. اگرداده ها را به صورت **5 کلاسه** درنظر بگیریم یعنی فرض کنیم فایل های صوتی موجود در دیتاست از 5 کلاس احساس مختلف تشکیل شده اند درنتیجه به دلیل کم بودن حجم داده های آموزش، دقت بسیار پایین خواهد بود. ( همانطور که از فایل خروجی قابل مشاهده است دقت داده های Validation بین 15 درصد تا 30 درصد متغیر خواهد بود درحالی که دقت بر روی داده های آموزش حتی به 50 درصد نیز می رسد. دلیل آن هم بخاطر کم بودن داده های آموزش می باشد که خاصیت تعمیم پذیری مدل را بشدت کاهش داده است و به اصطلاح Overfit بر روی داده های آموزش رخ داده است )
2. اگر به صورت **2 کلاسه** داده های خود را جمع آوری کنیم یعنی داده هایی را استخراج کنیم که تنها دو کلاس در بین آن ها وجود داشته باشد. در این صورت خواهیم دید که دقت داده های آموزش زمان آموزش بین 55 تا 70 و دقت داده های Validation نیز بین 60 تا 70 خواهد بود. به وضوح می توان دریافت زمانی که به دلایل مسایل سخت افزاری یا نرم افزاری نمی توانیم بر روی حجم زیادی از داده ها مدل را آموزش دهیم، هرچه تعداد کلاس ها را افزایش دهیم دقت نیز کاهش خواهد یافت که در شکل زیر خروجی حالت بیان شده را مشاهده می کنید

**نتیجه گیری :** زمانی که تعداد داده های درون دیتاست کم باشد، هرچه تعداد کلاس ها را افزایش دهیم دقت کاهش خواهد یافت. درنتیجه زمانی که تعداد کلاس های ما زیاد باشد (مانند دیتاست های موجود جهت پردازش سیگنال و صدا ) باید حجم دیتاست را نیز زیاد درنظر بگیریم. **به همین دلیل برای ارزیابی دیتاست “Berlin”، باید بر روی یک کامپیوتر قوی مدل را آموزش دهیم و سپس از مدل ذخیره شده استفاده کنیم تا عملیات پیش بینی را انجام دهیم.** حال باتوجه به توضیحات داده شده، مدل از پیش آموزش دیده شده ی موجود بر روی دیتاست “Berlin” را برای پیش بینی بر داده های تست دیتاست “Berlin” استفاده خواهیم نمود.

**پیش بینی نوع احساس موجود برای داده های جدید توسط مدل های آموزش دیده شده ( کد اول )**

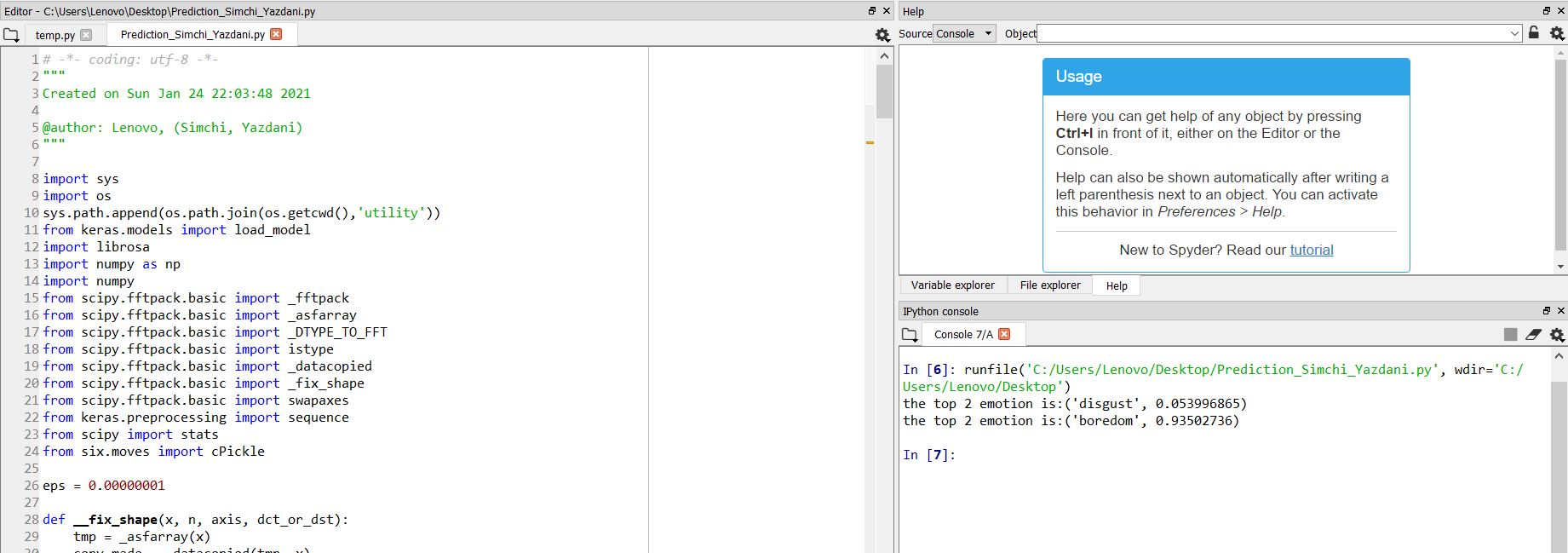
برای پیش بینی، از مدل از پیش آموزش دیده بر روی دیتاست “Berlin” استفاده می کنیم. برای پیش بینی احساس موجود کافی است درکنار توابع استخراج ویژگی بیان شده در قسمت های قبل، تابعی تعریف کنیم تا ماتریس Confusion را محاسبه کند. این ماتریس به ما کمک می کند تا درنهایت بتوانیم احساس با بالاترین درصد ممکن را بدست آورده و به عنوان احساس موجود به آن جمله نسبت دهیم.

همانطور که از شکل فوق مشخص است، دو تا از بالاترین احتمالات را برگردانده و تشخیص احساس باتوجه به برچسب داده ی وارد شده کاملا صحیح است ( داده ی وارد شده دارای برچسب عصبانی بوده است که فایل آن در پوشه ی Berlin\_Result قرار داده شده است )

**حال می خواهیم بررسی کنیم آیا می توان مدل را بر روی دیتاست “Berlin” آموزش داد و پیش بینی را بر روی دیتاست “ShEMO” انجام داد ؟**

\*\*\* باتوجه به اینکه برای هر داده برچسب نظیر آن را می دانیم درنتیجه باید قابل مقایسه باشد

درنتیجه، یکی از فایل های صوتی مربوط به دیتاست “ShEMO” را به مدل داده تا برچسب را برگرداند و سپس با مقایسه می توانیم به این سوال پاسخ دهیم.

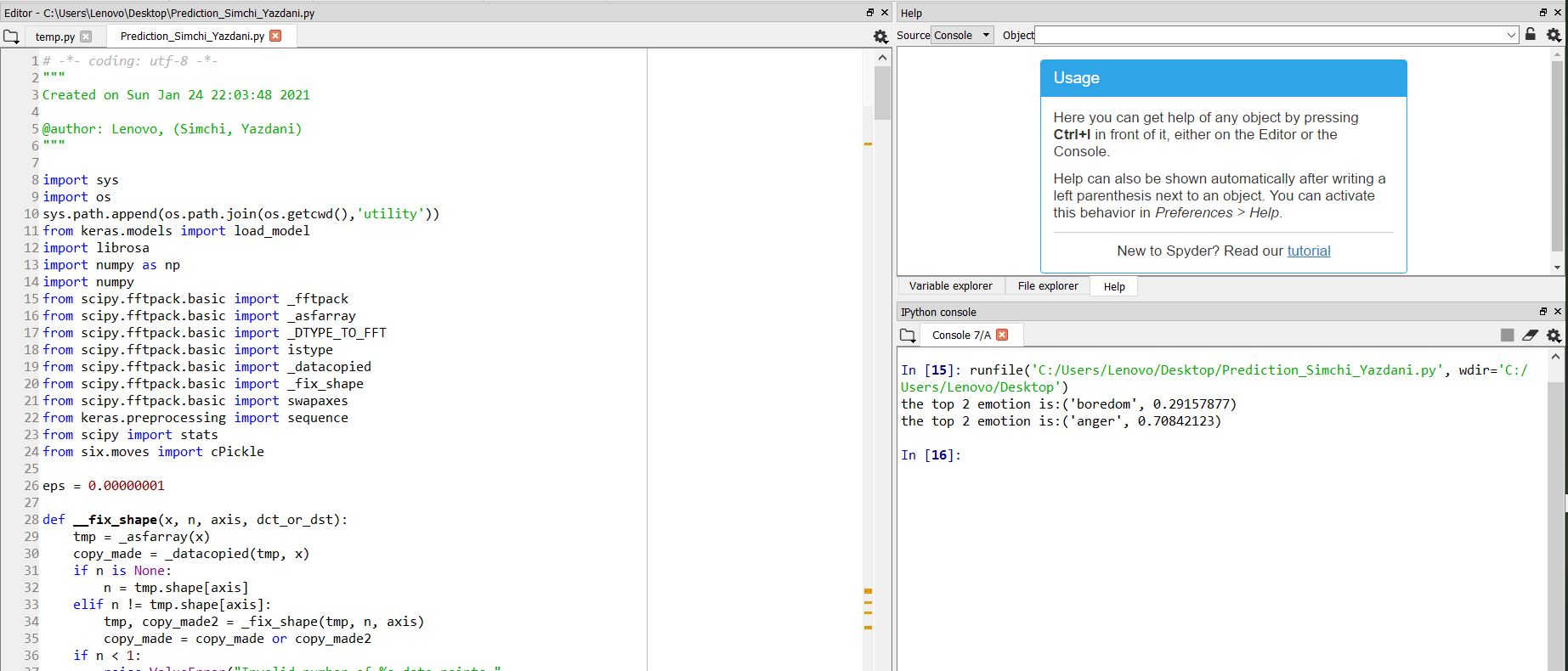
*پاسخ منفی است!!!*

همانطور که از شکل فوق مشخص است، برچسب های داده شده، انزجار و یا دلزدگی را بیان می کند درحالی که صدای داده شده نوعی عصبانیت را نشان می دهد.

**نتیجه گیری**

نوع احساس موجود در جملات و هر زبان بستگی به گویش همان زبان و نحوه ی بیان آن فرهنگ دارد.

حال می خواهیم بر روی مدل کوچک دو کلاسه ی آموزش دیده شده ی بر روی دیتاست “ShEMO” پیشگویی را انجام دهیم

درصورتی که فایل ورودی دارای یکی از برچسب های موجود در دیتاست زمان آموزش باشد، به درستی پیش بینی را انجام می دهد

با دقتی نزدیک به 78 درصد، صدای ورودی را عصبی تشخیص می دهد که کاملا صحیح است.

قسمت دوم

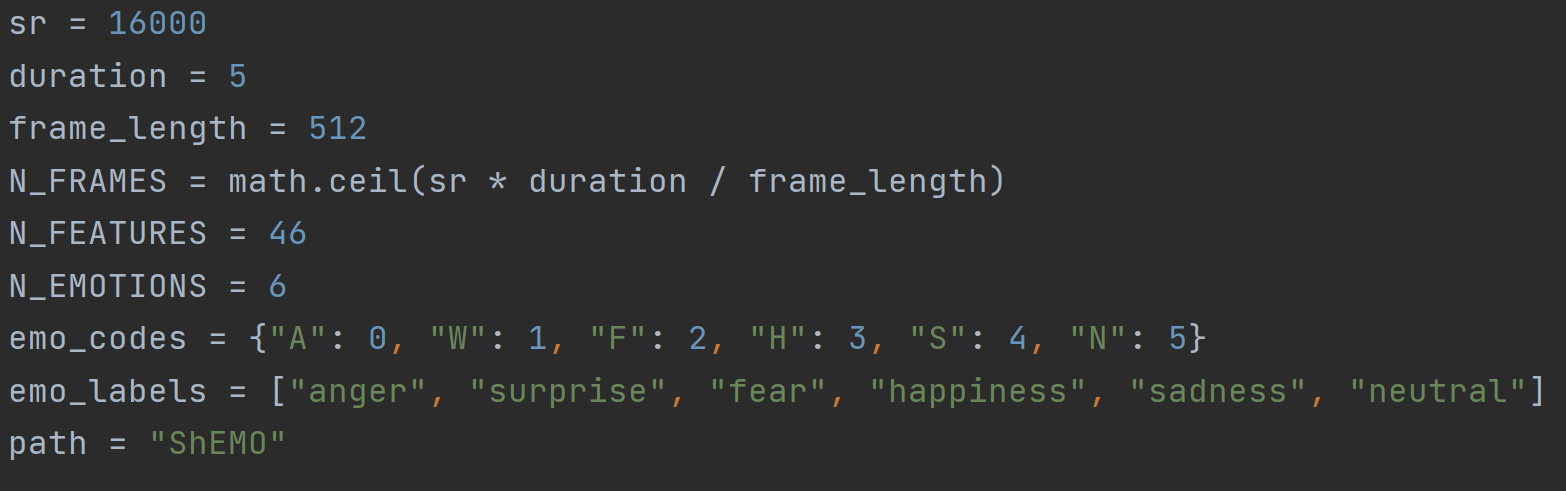
(بررسی کد دوم)

**کد دوم**

در ادامه، روش دوم پیاده سازی را بررسی خواهیم کرد. این روش شامل دو فایل مربوط به پیش پردازش داده ها و همچنین معماری شبکه عصبی مبتنی بر مکانیزم attention است، که به ترتیب در فایل های preprocessing.py و model.py پیاده سازی شده اند.

**بخش اول**

**پیش پردازش داده ها**

در این بخش ابتدا نیاز است مقادیر ثابتی که در طول اجرای الگوریتم مورد نیاز هستند را مشخص کنیم. به عنوان مثال می توان به فرکانس نمونه برداری که برابر با 16کیلوهرتز است اشاره کرد. لیست کامل این مقادیر در تصویر زیر مشخص شده است.

بهتر است مدت زمان هر فریم بین 20 تا 30 میلی ثانیه باشد، اما به دلیل محدودیت سخت افزاری این مقدار بر روی 512 تنظیم شده است. همچنین تعداد ویژگی هایی که از هر فریم استخراج خواهد شد، 46 عدد است.

**تابع feature\_extraction**

این تابع که به منظور استخراج ویژگی ها از سیگنال گفتار مورد نظر پیاده سازی شده است، خود دارای چند بخش میباشد.

1- خواندن فایل های صوتی و ذخیره سری زمانی آنها

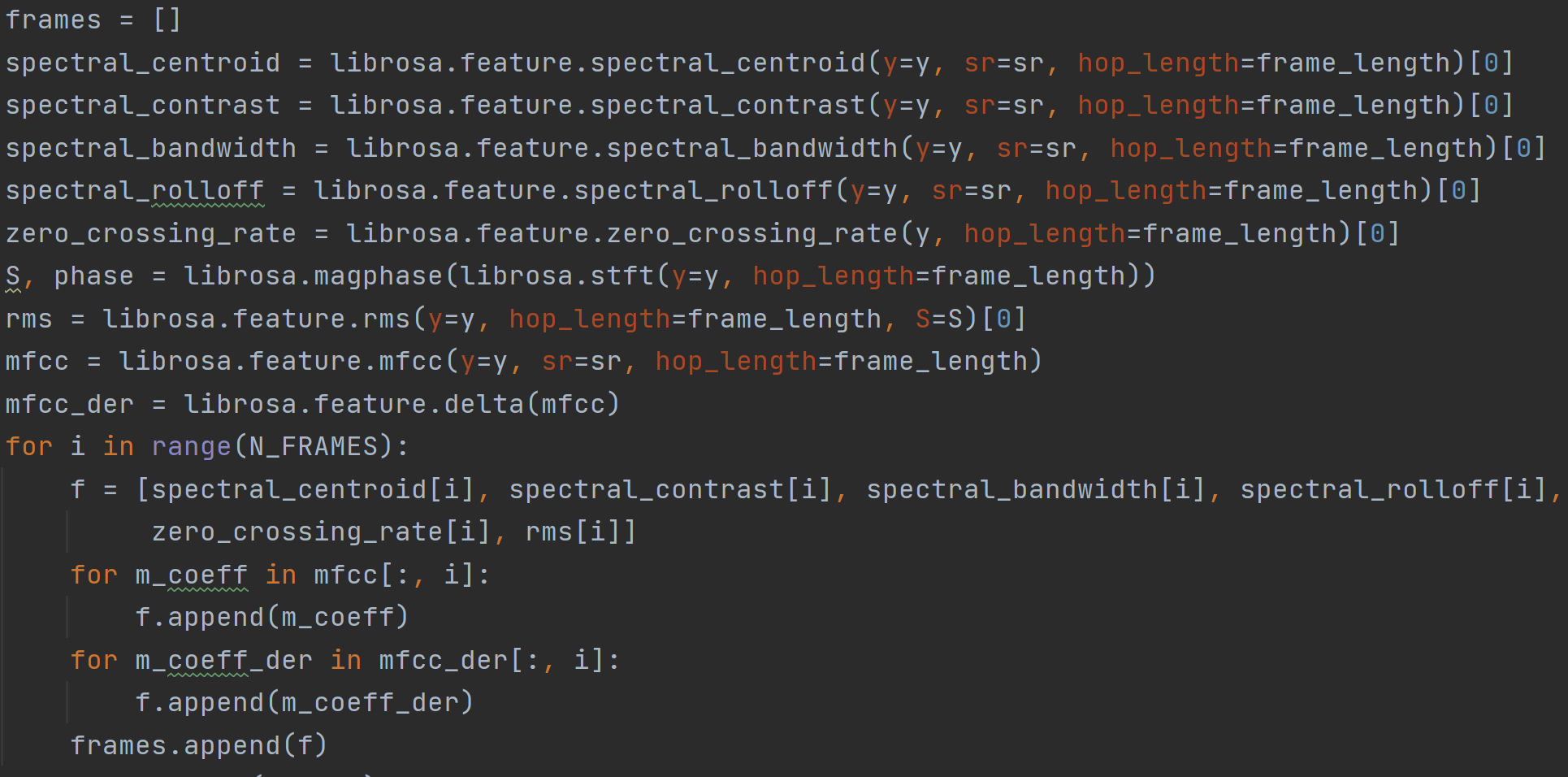
2- padding هر کدام از آنها در جهت یکسان شدن از نظر زمانی (به مدت زمان 5 ثانیه)

3- استخراج و ذخیره ویژگی های مربوط به هر فریمبه وسیله کتابخانه librosa

4- ذخیره و بازگرداندن بردار های ویژگی هر فریم و همچنین برچسب مورد نظر آنها به عنوان خروجی

**تابع get\_train\_test**

ابن تابع با توجه به بردارهای ویژگی استخراج شده و همچنین برچسب های ذخیره شده هر کدام از آنها در مرحله قبل، اقدام به ساختن مجموعه داده های آموزشی و تست کرده و هر یک را به عنوان خروجی بازمیگرداند. گفتنی است که برچسب های مورد نظر با روش one-hot encoding به صورت بردار هایی که فقط یکی از مقادیر آنها 1 و بقیه مقادیر 0 هستند درآمده است.

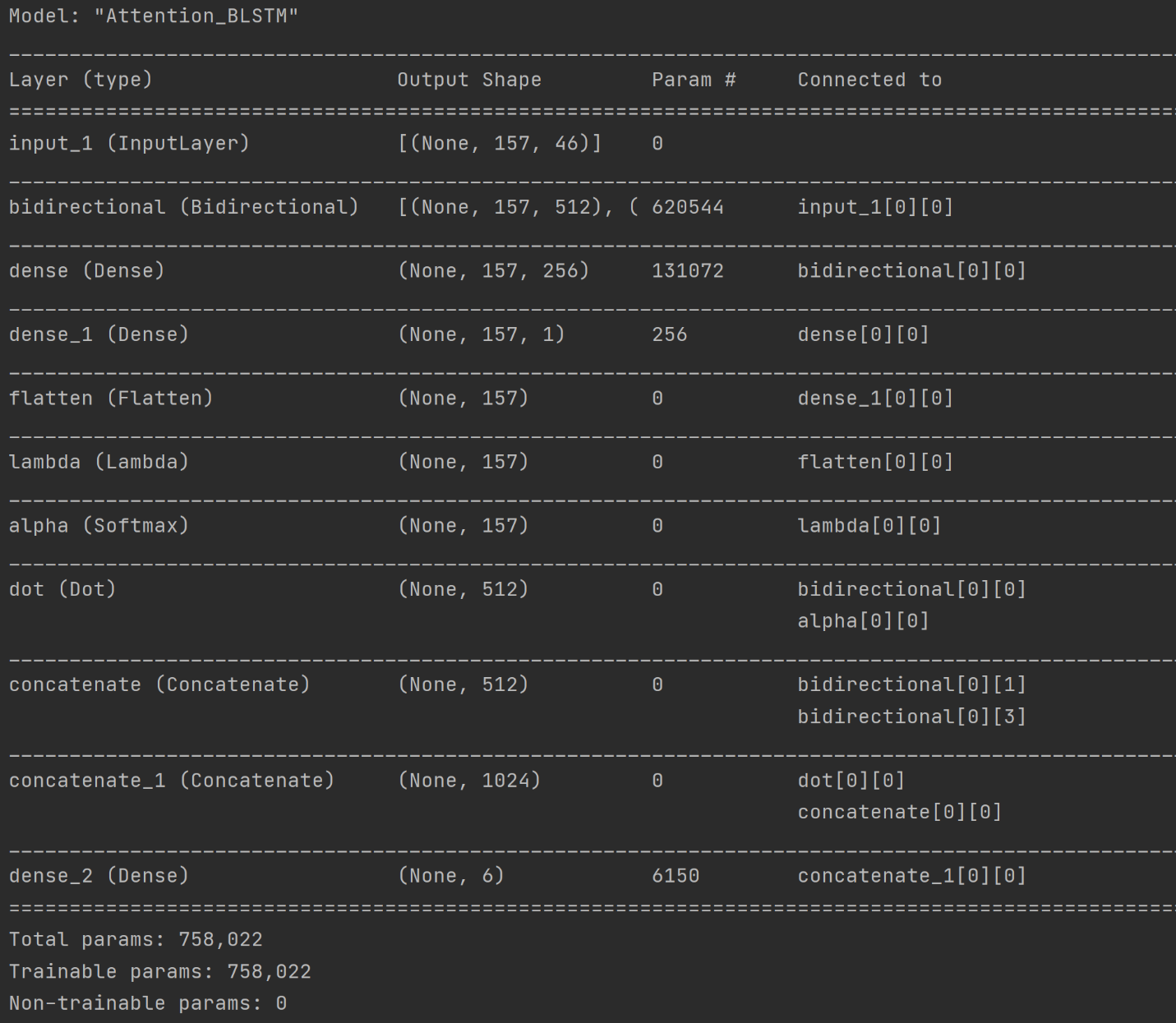
در ادامه تصویری از ویژگی های استخراج شده برای هر فریم و نحوه ذخیره آنها را خواهیم دید.

**بخش دوم**

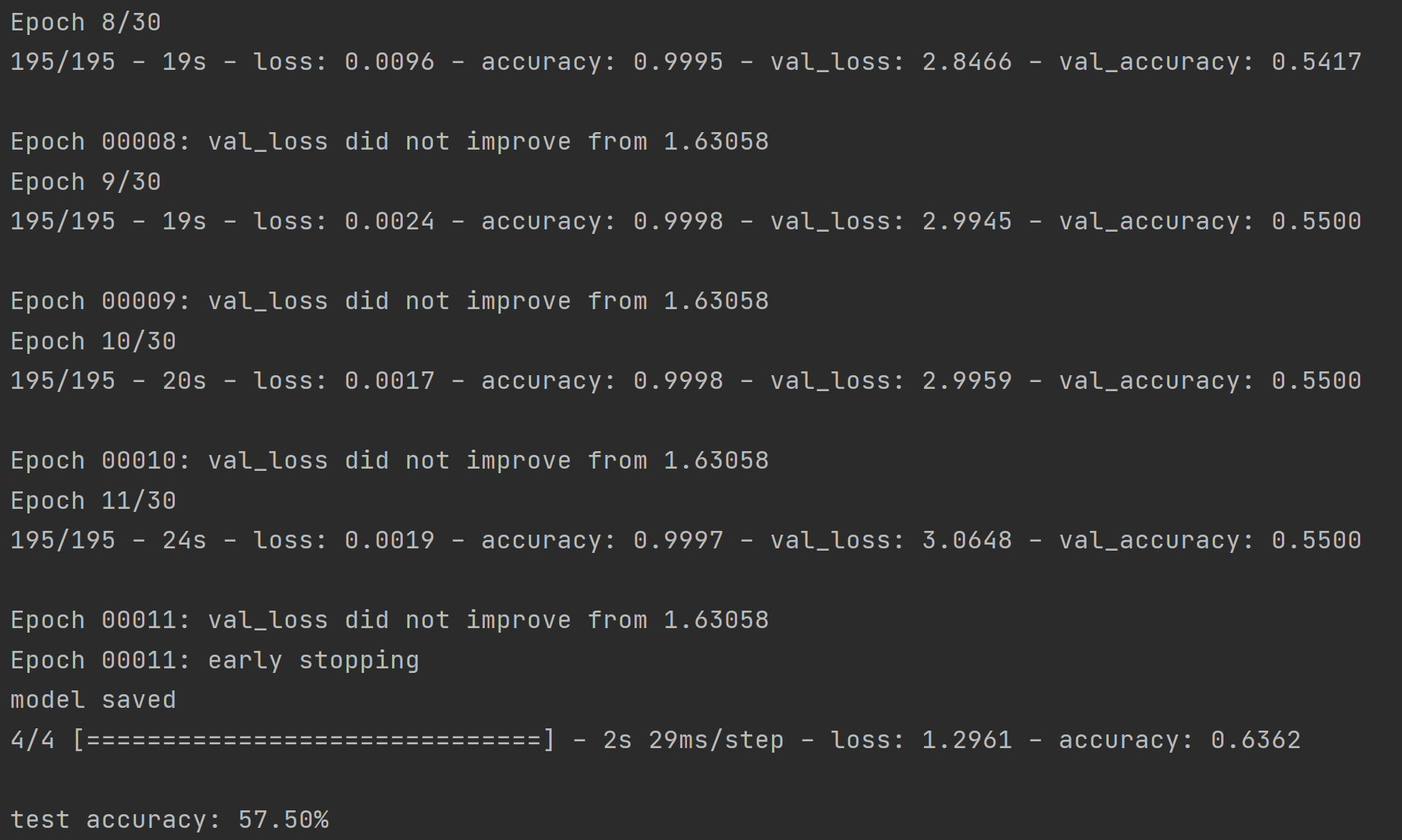
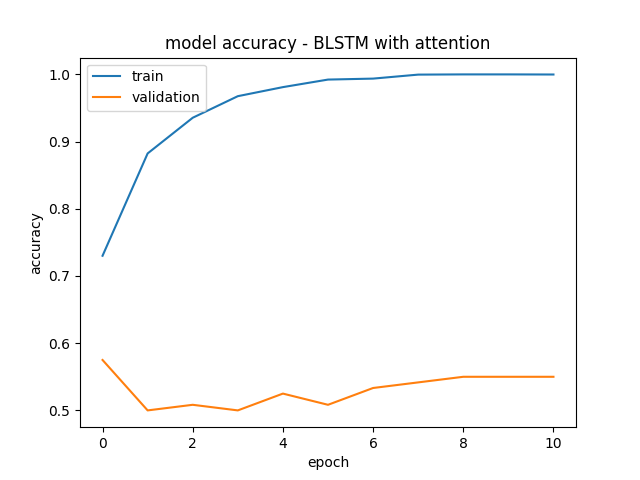
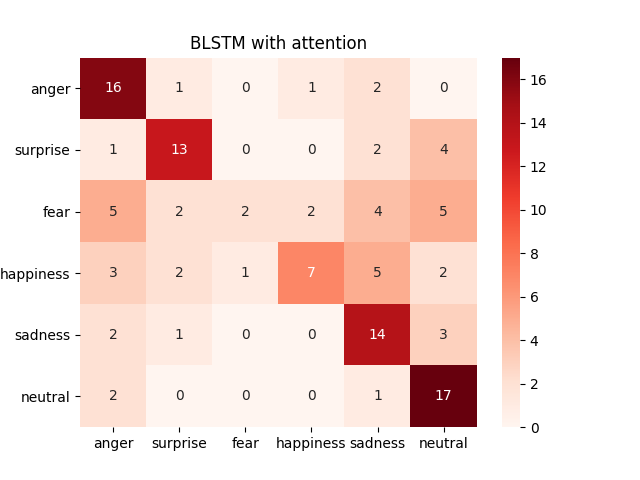
**پیاده سازی، آموزش و تست مدل**

پیاده سازی بخش دوم که مبتنی بر کتابخانه keras میباشد، شامل بخش های مختلفی از جمله ایجاد مدل مورد نظر، آموزش آن با توجه به مجموعه داده های آموزشی که در بخش قبل ایجاد شد و همچنین تست مدل با توجه به مجموعه داده تست است.

**تابع create\_model**

به وسیله این تابع، مدل مورد نظر خود که از مکانیزم توجه استفاده میکند را، ایجاد میکنیم. تعداد واحدها یا نورون­های هر لایه به طور پیش فرض، 256 عدد در نظر گرفته شده است. سایر جزئیات این مدل به شرح زیر است:

**تابع train\_and\_test\_model**

در این تابع، مجموعه داده های آموزشی و تست که در بخش اول تعبیه شد را روی مدل مورد نظر آموزش داده و تست میکنیم. برای آموزش مدل از بهینه ساز Adam و همچنین روش Early Stopping برای جلوگیری از بیش برازش استفاده شده است. در نهایت نمودار دقت و ماتریس کوواریانس مربوط به این آزمایش در تصاویر زید آمده است.

در تصویر زیر نیز بخشی از مراحل آموزش مدل و همچنین دقت نهایی مدل روی مجموعه داده ShEMO مشخص شده است.

قسمت سوم

(شرح نحوه ی اجرای کد اول )